

STREAMING

UN STREAM ES UN FLUJO DE DATOS QUE NO TIENE FIN. LOS ALGORITMOS TRADICIONALES TIENEN GRAVES PROBLEMAS AL PROCESAR STREAMS. ESTO ES PORQUE ES MUY DIFÍCIL CALCULAR ESTADÍSTICAS, RANKINGS O METRÍCAS SI LOS DATOS CAMBIAN CONSTANTEMENTE, ESTO SE DEBE A CUANDO CALCULAMOS CIERTO VALOR, ESTE YA PERDIÓ ACTUALIDAD POR LA CANTIDAD DE NUEVA INFO RECIBIDA.

LOS ALGORITMOS DE STREAMING ESTÁN DISEÑADOS PARA RESOLVER ÉSTOS PROBLEMAS. ESTOS TRABAJAN EN MEMORIA Y PROCESAN LOS DATOS A MEDIDA QUE ESTOS OCURBEN ACTUALIZANDO SU RESULTADO. ESTO NOS PERMITE OBTENER EN TIEMPO REAL DATOS SOBRE EL STREAM, PERMITIENDONOS SABER QUE ESTÁ OCURRIENDO Y DEJANDONOS TOMAR DECISIONES.

RESERVOIR SAMPLING

ESTE PRIMER ALGORITMO NOS PERMITE CONVERTIR UN STREAM INFINITO EN UN CONJUNTO DE DATOS FINITOS. LA IDEA ES TENER EN CADA MOMENTO UNA MUESTRA DEL STREAM EN MEMORIA. ASÍ PODEMOS CALCULAR DATOS O ESTADÍSTICAS UTILIZANDO ALGORITMOS TRADICIONALES. EL ALGORITMO DEBE CUMPLIR:

- DEBE MANTENER EN MEMORIA UNA CANTIDAD FIJA DE ELEMENTOS K.
- DEBE GARANTIZAR QUE LA PROBABILIDAD DE QUE UN DATO DEL STREAM ESTÉ EN LA MUESTRA

$$P(X_i \in \text{MUESTRA}) = \frac{k}{n}$$

DONDE N ES LA CANTIDAD DE ELEMENTOS DEL STREAM HASTA EL MOMENTO.

- LA PROBABILIDAD ES DINAMICA, A MEDIDA QUE OBSERVAMOS MAS DATOS LA PROBABILIDAD BAJA.
- LA PROBABILIDAD DE TODOS LOS ELEMENTOS INDEPENDIENTEMENTE DE CUANDO LOS OBSERVEMOS DEBE SER IGUAL.

EL ALGORITMO

POB CADA DATO QUE GENERA NUESTDO STREAM CALCULAMOS $P = k/n$ (LA IP DE QUE EL DATO INGRESSE INGRESÉ A LA MUESTRA).

LUEGO, GENERAMOS UN NUMERO RANDOM $\in (0-1)$. Y DE ACUERDO A P DECIMOS SI EL DATO INGRESA A LA MUESTRA O NO.

- Si $P > n.r \rightarrow$ INGRESA
- Si $P < n.r \rightarrow$ NO INGRESA.

SI EL DATO INGRESA, ELEGIMOS AL AZAR UN DATO DE LA MUESTRA Y LO REEMPLAZAMOS POB EL NUEVO.

MOMENTOS DE UN STREAM

UN STREAM ES UNA COLECCIÓN DE ELEMENTOS, LLAMEMOS m_i A LA CANTIDAD DE VECES QUE EL ELEMENTO i OCURRIDO EN EL STREAM (FRECUENCIA)

$$M^k(S) = \sum_{i \in S} (m_i)^k$$

MOMENTO DE ORDEN K

- **MOMENTO DE ORDEN 0**: ES LA CANTIDAD DE ELEMENTOS DISTINTOS EN EL STREAM.

CALCULAR ESTE MOMENTO NO ES SENCILLO YA QUE POB CADA ELEMENTO DEL STREAM NECESETAMOS SABER SI ES NUEVO O NO, TAMPOCO PODEMOS TENER TODOS LOS ELEMENTOS EN MEMORIA Y COMPARAR.

- 2
- **MOMENTO DE ORDEN 1**: LA CANTIDAD DE ELEMENTOS EN TOTAL EN EL STREAM, ES N .
 - **MOMENTO DE ORDEN 2**: SE LO CONOCE COMO NUMERO SOPRESA DEL STREAM. ES UN INDICADOR DE SI LOS DATOS DEL STREAM SE DISTRIBUYEN DE MANERA PARAJA O SI ALGO PREDOMINA.

FLAJOLET - MARTÍN

ESTE ALGORITMO NOS PERMITE CALCULAR LA CANTIDAD DE ELEMENTOS DISTINTOS OBSERVADOS HASTA EL MOMENTO EN UN STREAM, ES DECIR, EL MOMENTO θ .

ESTE ALGORITMO USA UNA FUNCIÓN DE HASHING Y UN CONTADOR EN MEMORIA. POR CADA DATO QUE OBSERVAMOS EN EL STREAM LE APPLICAMOS $H(x)$ Y CONTAMOS LA CANTIDAD DE $0s$ CONSECUTIVOS A PARTIR DEL BIT 0 . EN MEMORIA ESTA Q , CON LA CANTIDAD MÁXIMA DE $0s$ QUE HEMOS OBSERVADO AL COMIENZO.

ENTONCES EL MOMENTO DE ORDEN θ SE ESTIMA COMO

$$M^{\theta}(S) \approx 2^R$$

POB LO TANTO CALCULAR 2^R APPROXIMA A LA CANTIDAD DE ELEMENTOS QUE VIMOS EN EL STREAM.

SIN EMBARGO, PUEDE SUCEDER QUE ALCUNOS DATOS COLISIONEN. ESTO HACE AL ALGORITMO MUY SENSIBLE A LOS VALORES DE LA FUNCIÓN DE HASHING.

UNA FORMA DE SOLUCIONAR ESTO ES:

- SE DIVIDEN LOS k ESTIMADORES EN b GRUPOS DE m ESTIMADORES CADA UNO Y SE CALCULA COMO RESULTADO FINAL EL PROMEDIO DE LAS MEDIAS DE CADA GRUPO.

HYPERLOGLOG

ESTE ALGORITMO ES UNA VERSION MEJORADA DE LOS MISMOS ALGORITMOS Y QUE ES LA QUE HOY EN DIA MAS SE USA PARA AUTODES Y QUE ESTIMA LA CANTIDAD DE ELEMENTOS EN UN STREAM.

ESTE USA UNA UNICA FUNCION DE HASHING Y SOBRE LA MISMA CONSTRUYE VARIOS ESTIMADORES, CALCULANDO EL RESULTADO FINAL COMO EL PROMEDIO ARMONICO DE LOS ESTIMADORES.

LA H(X) NOS GENERA UN NUMERO DE 64 BITS, LOS PRIMEROS K BITS LOS VAMOS A USAR PARA EL NUMERO DE ESTIMADORES. DE LOS RESTANTES BITS CONTAMOS COMO SIEMPRE LA CANTIDAD DE Ceros AL COMIENZO DE LOS MISMOS Y ACTUALIZAMOS EL CONTADOR DEL ESTIMADOR SI ESTA CANTIDAD ES MAYOR A LA ANTERIOR.

$$M^o(S) \approx \frac{N}{\sum_{j=1}^n 2^{-R_j}}$$

EL USO DEL PROMEDIO ARMONICO SE DEBE A QUE ESTE MITIGA EL EFECTO DE VALORES MUY GRANDES EN EL PROMEDIO Y AUMENTA EL EFECTO DE VALORES PEQUEÑOS.

AMS

ESTE ALGORITMO SIRVE PARA ESTIMAR EL MOMENTO DE ORDEN 2 DE UN STREAM; ESTE ALGORITMO MANTIENE EN MEMORIA K ESTIMADORES, CADA ESTIMADOR REGISTRA UN ELEMENTO DEL STREAM Y LA CANTIDAD DE VECES QUE APARECIO EL MISMO. POR CADA ELEMENTO, SI EL MISMO ESTA DENTRO DE LOS ESTIMADORES AUMENTAMOS EN UNO EL CONTADOR, CASO CONTRARIO INGRESA A MEMORIA CON PROBABILIDAD K/N, SI EL ESTIMADOR INGRESA A LA MEMORIA REEMPLAZA A OTRO ESTIMADOR AL AZAR.

CADA ESTIMADOZ ESTIMA EL MOMENTO DE ORDEN 2 3

$$M^2(S) \approx N(2C_i - 1)$$

SIENDO C_i EL CONTADOR Y N LA CANTIDAD DE ELEMENTOS DEL STREAM.

VAMOS A AGRUPAR LOS ESTIMADORES EN B GRUPOS DE M ESTIMADORES CADA UNO SIENDO EL RESULTADO FINAL LA MEDIANA DEL PROMEDIO DE CADA GRUPO.

EJEMPLO

TENEMOS EL STREAM $S = [1 2 3 2 4 2 5 3 4 3 1]$

ESTE STREAM TIENE 12 ELEMENTOS ($N = 12$)

AHORA VAMOS APLICAR AMS CON $K=3$ Y 1 ESTIMADOR

OBS	K1	C1	K2	C2	K3	C3
1	1	1				
2	1	1	2	1		
3	1	1	2	1	3	1
2	1	1	2	2	3	1
4	1	1	2	2	3	1
2	1	1	2	3	3	1
5	1	1	2	3	5	1
3	1	1	2	3	5	1
4	4	1	2	3	5	1
1	4	2	2	3	5	1
3	4	2	2	3	5	1
1	4	2	2	3	5	1

EMPIEZO COLOCANDO EL PRIMER ELEMENTO EN EL PRIMER G. GRUPO. ASÍ VOY AVANZANDO.

→ ESTE SE ENCUENTRA EN K_2 ASÍ QUE INCREMENTO 1.

→ AHORA OBSERVAMOS QUE 4 NO ESTÁ EN NINGUN K. ENTONCES APLICAMOS RESEÑA DE SAMPUNG.

• HACEMOS UNA ESPECIE DE SOBTEO Y VEMOS SI 4 ENTRA O NO.

Y ASÍ CONTINUA EL DESTO DEL ALGORITMO.

AHORA PODEMOS ESTIMAR $M^2(S)$ CON $N(2C_i - 1)$ PARA CADA GRUPO.

$$k_1 = 12 * (2 * 2 - 1) = 36$$

$$k_2 = 12 * (2 * 3 - 1) = 60$$

$$k_3 = 12 * (2 * 1 - 1) = 12$$

→ PARA LOS VALORES DE c_i SIEMPRE SON LOS DE LA ULTIMA FILA.

ADEMÁS TENÉS EN CUENTA QUE ACA TENGO UN ESTIMADOR DE GRUPO, SI CADA GRUPO TUVIESE MAS, HAGO EL PROMEDIO DE LOS VALORES.

FINALMENTE TOMAMOS LA MEDIANA DE LOS PROMEDIOS DE CADA GRUPO.

POD LO QUE $M^2(S) = 36$ | $M^2(S) = 2^2 + 3^2 + 3^2 + 3^2 + 1^2 = 32$
IDEAL
LOS RESULTADOS SON SIMILARES

FILTROS DE BLOOM

DADO UN STREAM QUEREMOS SABER SI LOS ELEMENTOS QUE OBSERVAMOS PERTENECEN O NO A UN CIERTO CONJUNTO DE ELEMENTOS PREDEFINIDOS.

ENTONCES UN FILTRO DE BLOOM, ES UN VECTOR BINARIO DE M BITS Y k FUNCIONES DE HASHING.

PARA AGREGAR UN ELEMENTO AL FILTRO LE APLICAMOS LAS FUNCIONES DE HASHING Y ENCENDEMOS LOS BITS QUE QUEDAN APUNTADOS POR LAS FUNCIONES. SE PREDEN k O MENOS BITS SEGUN HAYA COLISIONES O NO.

PARA VERIFICAR SI UN ELEMENTO PERTENECE A NUESTRO CONJUNTO, LE APLICAMOS LA $H(x)$ Y VERIFICAMOS SI TODOS LOS BITS ESTAN ENCENDIDOS, CASO CONTRARIO SI HAY AL MENOS UN BIT EN 0, EL ELEMENTO NO PERTENECE.

RECORDAD QUE SI EL ELEMENTO PERTENECE, ESTO SUcede CON UNA CIERTA PROBABILIDAD YA QUE PODRIA HABER FALSOS POSITIVOS.

RECORDANDO QUE TENEMOS K FUNCIONES DE HASH, N ELEMENTOS EN EL STREAM Y M BITS

- LA PROBABILIDAD DE QUE UNA FUNCIÓN ENCIENDA UN CIEGTO BIT ES $1/M$.
- LA PROBABILIDAD DE QUE UNA FUNCIÓN NO ENCIENDA UN CIEGTO BIT ES $1 - (1/M)$
- LA PROBABILIDAD DE QUE NINGUNA FUNCIÓN ENCIENDA UN CIEGTO BIT ES $(1 - e^{KN/M})^K$

PARA VALORES DE N Y M FIJOS:

$$K = \frac{M}{N} \ln(2)$$

VALOR ÓPTIMO DE K

$$M = -\frac{N \ln(P)}{(\ln(2))^2}$$

VALOR ÓPTIMO DE M

$$E \approx \frac{M \ln(1 - \frac{x}{M})}{K}$$

CON ESTO PODEMOS ESTIMAR LA CANTIDAD DE ELEMENTOS QUE HAN SIDO INSERTADOS EN EL FILTRO.

CANTIDAD DE BITS EN 1

→ SI QUEREMOS PODER INSERTAR Y ELIMINAR ELEMENTOS DEL FILTRO ENTONCES ESTA ESTRUCTURA NO SIRVE PORQUE NO PODEMOS AL BORRAR APAGAR LOS BITS EN 1 INDICADOS POR LAS K FUNCIONES DE HASHING PUES ESTOS BITS PODRÍAN CORRESPONDER A OTROS ELEMENTOS.

COUNTING FILTER

ES LA SOLUCIÓN YA QUE EN LUGAR DE PONER 1 BIT POR CADA POSICIÓN TIENE UN ENTEDO DE F BITS, ASÍ SE PUEDE INCREMENTAR O DECREMENTAR EL VALOR. ESTO OCUPA MUCHA MEMORIA

CUCKOO FILTERS

ESTE ES UNA VARIANTE A LOS FILTROS DE BLOOM. ESTE MÉTODO UTILIZA CUCKOO HASHING COMO ESTRUCTURA PRINCIPAL. LAS DIFERENCIAS PRINCIPALES CON BLOOM SON:

- REQUIEREN MENOS ESPACIO
- SON MAS RÁPIDOS EN CONSULTAS.
- SON MAS LENTOS EN INSECCIONES.
- PERMITEN LA OPERACIÓN DE BORRADO.

PARA EMPEZAR, EL CUCKOO FILTER NO ALMACENA LAS CLAVES PORQUE ESTO HABIA QUE DEPENDA DE ELLAS Y NO SERIA COMPACTO. ESTE ALMACENA **UN HASH DE LA CLAVE**, LLAMADO **FINGERPRINT**. (ESTOS OCUPAN ENTRE 6 Y 8 BITS).

LA TABLA DE CUCKOO VA A TENER M BUCKETS EN DONDE CADA BUCKET ALMACENA B F.P.

ADEMÁS NECESITAMOS DOS POSICIONES ALTERNATIVAS PARA LAS CLAVES:

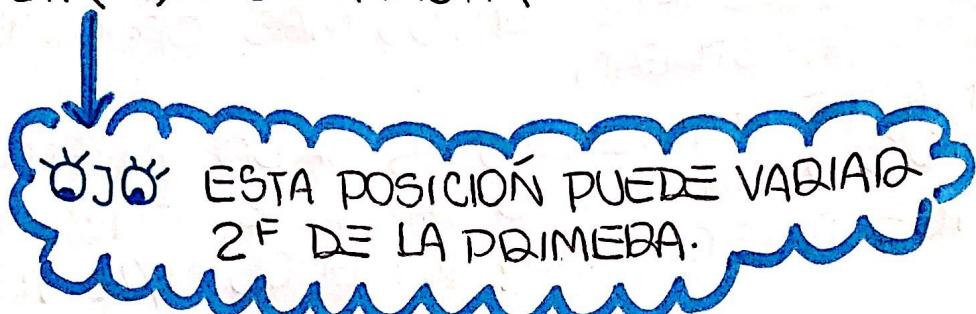
- **PRIMERA POSICIÓN**: SE APLICA UNA $H(x)$ A LA CLAVE. (NO AL F.P.)

SI EL BUCKET TIENE EL LUGAR LIBRE, CALCULAMOS EL F.P DE LA CLAVE Y LO ALMACENAMOS EN EL BUCKET.
SI EL BUCKET NO ESTA LIBRE:

- **SEGUNDA POSICIÓN**: GUARDAMOS EL F.P DE LA CLAVE EN ESTE BUCKET (DADO POR $H(x)$) Y ALEATORIAMENTE MOVEMOS OTRO F.P

TENEMOS UN PROBLEMA ACA, NO PODEMOS APLICAR UNA $H_2(x)$ PORQUE NO TENEMOS LA CLAVE, SOLO EL F.P. ENTONCES COMO SOLUCIÓN A ESTO, SE LE APLICA LA $H_2(x)$ AL F.P Y SE HACE UN XOR ENTRE LA POSICIÓN ORIGINAL Y EL HASH DEL F.P. POR LO TANTO, LAS POSICIONES ALTERNATIVAS SON:

- $\text{HASH}(x)$
- $\text{HASH}(x) \oplus \text{HASH}(\text{F.P.})$



COMO EN CUCKOO LAS INSERCCIONES PUEDEN FALLAR LA CANTIDAD DE FALSOS POSITIVOS QUEDA ACOTADA INDEPENDIENTEMENTE DE LA CANT DE DATOS QUE INSERTEMOS.

OTRA PROPIEDAD INDESEABLE, ES QUE ESTE SOPORTA EL BORRADO. SI QUEREMOS ELIMINAR UNA CIERTA CLAVE OBTENEMOS SUS DOS POS ALTERNATIVAS, Y SI ENCONTRAMOS EL F.P ELIMINAMOS LA COPIA Y LISTO.

ENTONCES, PARA CUCKOO ES NECESARIO

- N = CANTIDAD DE CLAVES
- M = CANTIDAD DE BUCKETS
- B = CANTIDAD DE REGISTROS POR BUCKET
- F = TAMAÑO DEL FINGERPRINT.
- LA PROBABILIDAD DE FALSOS POSITIVOS = $81/2^F$
SIENDO λ EL FACTOR DE CARGA ($\lambda = N/BM$)
ENTONCES AUMENTANDO F, REDUCIMOS LA CANTIDAD DE FALSOS POSITIVOS A COSTA DE AUMENTAR EL TAMAÑO DE LA ESTRUCTURA.

COUNT-MIN

ESTE ALGORITMO PERMITE PARA CUALQUIER DATO DE UN STREAM ESTIMAR CUANTAS VECES OCURRIÓ EL MISMO HASTA EL MOMENTO. ESTO NOS PERMITE RESOLVER EL PROBLEMA DE HEAVY HITTERS.

HEAVY HITTERS

ESTE PROBLEMA CONSISTE EN ENCONTRAR LOS ELEMENTOS MÁS POPULARES EN UN STREAM. UNA FORMA DE PLANTEARLO ES DADO UN STREAM DE N ELEMENTOS Y UN CIERTO VALOR K , ENCONTRAR LOS ELEMENTOS QUE OCURRAN AL MENOS N/K VECES EN EL STREAM.

LAMENTABLEMENTE NO EXISTE ALGORITMO QUE DESUELVA EL HH PROBLEMA EN $O(N)$ USANDO UNA CANTIDAD SUBLINEAL DEL ESPACIO AUXILIAR. ENTONCES SI QUEDEMOS RESOLVER ESTE PROBLEMA, VAMOS A TRABAJAR CON UNA APROXIMACIÓN.

HH APROXIMADO

EN ESTE PROBLEMA APROXIMADO TENEMOS UN STREAM S DE N ELEMENTOS Y DADO UN K QUEREMOS CUMPLIR:

- TODO VALOR QUE OCURRE AL MENOS N/K VECES EN S ESTÁ EN EL RESULTADO
- TODO VALOR EN EL RESULTADO OCURRE AL MENOS $N/K - \epsilon$ VECES EN S .
- EL ESPACIO USADO ES $O(1/\epsilon) = O(2K)$

COUNT-MIN SKETCH

ESTE ALGORITMO ES EQUIVALENTE A TENER L COUNTING FILTERS. NECESITAMOS L FUNCIONES DE HASHING (SIENDO B SU ESPACIO DE DIRECCIONES).

POD CADA DATO DEL STREAM, APLICAMOS LAS L FUNCIONES DE HASHING E INCREMENTAMOS EL BUCKET APUNTADO POR LAS FUNCIONES EN CADO UNO DE LOS FILTROS. LA ESTIMACIÓN DEL ALGORITMO SOBRE LA CANTIDAD DE VECES QUE APARECIO UN ITEM ES EL MINIMO DE LOS L CONTADORES.

EL PROBLEMA DE LOS TOP K ELEMENTOS ES SENCILLO, NECESITAMOS UNA TABLA DE K ELEMENTOS, Y CADA VEZ QUE OCURRE UN ELEMENTO ESTIMAMOS SU CARDINALIDAD Y ACTUALIZAMOS EL RANKING.

EJEMPLO

CONSTRUCCIÓN DE UN COUNTING FILTER

	A	B	C	D
H1	12	8	4	4
H2	4	6	9	15

ESTOS NUMEROS NOS INDICAN QUÉ BITS TENEMOS QUE PRENDER (IZQ A DERECHA)

A DIFERENCIA DE LOS BLOOM, SI ACA SE REPITE UN NUMERO LE SUMAMOS UNO AL BIT

[0000 3010 1100 1001] → FILTRO DE COUNTING
 ↳ EL CUATRO APARECE TRES VECES.

AHORA PARA COUNT MIN, TENEMOS L FILTROS DE COUNTING

[5 146 3428] Y CON ESTOS QUEDAMOS ESTIMAR LA FRECUENCIA DE A.
 [3 254 1726]
 [6 315 3274]

HASHEAMOS EL ELEMENTO A CON CADA H(X) Y OBTENEMOS LAS POSICIONES DE LOS CONTADORES

$$\left. \begin{array}{l} H_1(A) = 3 \\ H_2(A) = 2 \\ H_3(A) = 7 \end{array} \right\} \text{DE ESTOS ME QUEDO } (6, 5, 4) \text{ CON EL MIN } \min(6, 5, 4) \\ \text{FREC}(A) = 4$$